

Numpy :- supports for arrays and matrices.

↳ `import numpy as np.`

`arr1 = np.array([1,2,3,4,5])`

`print(arr1) → [1,2,3,4,5]`

`print(arr1.shape) → (5,)`

`arr2 = np.array([[1,2,3,4,5], [2,3,4,5,6]])`

`print(arr2) = [[1,2,3,4,5]`

`[2,3,4,5,6]]`

`np.arange(0,10,2).reshape(5,1) → [0]`

`[2]`

`[4]`

`np.ones((3,4)) → [[1., 1., 1., 1.], [1., 1., 1., 1.], [1., 1., 1., 1.]]`

`[6]`

`[8]`

`[1., 1., 1., 1.]]`

#identity matrix → `np.eye(3)`

↳ `array[[1., 0., 0.], [0., 1., 0.], [0., 0., 1.]]`

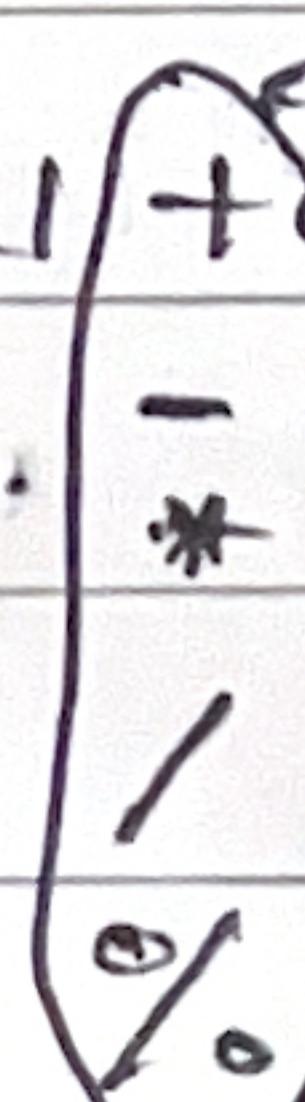
`arr.ndim → No. of dimensions = 2`

`[0., 0., 1.]`

`arr.size → No. of elements = 9`

Arithmetic operations :- `arr1 + arr2`

↳ for vector based



anything

Universal functions:-

`arr = np.array[2,3,4,6,8]`

`np.sqrt(arr) →`

`np.exp(arr) →`

`np.sin(arr) →`

`np.log(arr) →`

array slicing & indexing

```
arr = np.array([1,2,3,4], [5,6,7,8], [9,10,11,12])
```

```
print(arr[1:][2:]) → ([7,8])
```

```
print(arr[0][0]) → 1 [11,12])
```

modify arr[0][0] = 100 # replace 1 with 100

$$\text{arr}[0,0] = 100$$

Statistical concepts - Normalization

to have mean of 0 & s.d as 1:

```
data = np.array[ ]
```

```
mean = np.mean(data)
```

```
std-dev = np.std(data)
```

normalized_data = (data-mean) / (std-dev)

$\text{np.median}(\text{data}) \rightarrow$

$\text{np.variance}(\text{data}) \rightarrow$

performs logical operations:-

```
data = np.array([1,2,3,4,5,6,7,8])
```

```
data[(data >= 5) & (data <= 8)]
```

$$\hookrightarrow [5,6,7,8]$$

AND, OR, XOR, NOT

for arr1, arr2 to concatenate

$\text{arr3} = \text{np.concatenate}[\text{arr1}, \text{arr2}]$